DOCUMENT RESUME

ED 388 291 IR 017 440

AUTHOR Chee, Yam San

TITLE SMALLTALKER: A Cognitive Apprenticeship Multimedia Environment for Learning Smalltalk Programming.

PUB DATE 94

NOTE 7p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.

PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.

DESCRIPTORS *Computer Software; Feedback; Foreign Countries; *Multimedia Instruction; Programming; Student Interests; Student Motivation; *Teaching Methods

IDENTIFIERS *Cognitive Apprenticeships; Knowledge Acquisition; Learning Environment; *SmallTalk Programming Language

ABSTRACT

As an instructional technology, cognitive apprenticeship has become increasingly important. Cognitive apprenticeship embeds the acquisition of knowledge and skills in their social and functional context and consists of six teaching methods: modelling, coaching, scaffolding, articulation, reflection, and exploration. SMALLTALKER is a Macintosh-based multimedia learning environment for Smalltalk programming that supports both concept learning and skill acquisition. The system fulfills the roles of instruction presenter and coach. In its capacity as instruction presenter, SMALLTALKER makes extensive use of modelling; animations ar(used to present instruction on concepts and skills necessary for Smalltalk programming. The most pervasive method of coaching takes the form of feedback to student actions and errors while they are engaged in solving programming problems. Scaffolding and fading are difficult teaching methods to implement because they require a teacher to be sensitive to the needs and difficulties of students engaged in task performance; the SMALLTALKER approach to this difficulty is to place the burden of responsibility on the student and an open-ended help system is provided. SMALLTALKER poses conceptual questions to provoke both articulation and reflection. Exploration is an activity that system designers cannot prevent students from engaging in, as long as the system succeeds in gripping their interest and arousing their motivation to learn. (Contains 15 references.) (AEF)

SMALLTALKER: A Cognitive Apprenticeship Multimedia Environment for Learning Smalltalk Programming

YAM SAN CHEE
Department of Information Systems and Computer Science
National University of Singapore, Lower Kent Ridge Road, Singapore 0511
E-Mail: cheeys@iscs.nus.sg

Abstract: Cognitive apprenticeship has become increasingly important as an instructional methodology due to a shift by many researchers to the situated cognition paradigm. We describe *SMALLTALKER*, a learning environment that makes extensive use of multimedia to support the learning of Smalltalk programming. The design of *SMALLTALKER* is based on the principle of cognitive apprenticeship. We illustrate how modeling, coaching, scaffolding and fading, articulation, reflection, and exploration have been implemented in *SMALLTALKER*.

In recent years, cognitive apprenticeship (Brown, Collins, & Duguid, 1988; Burger & DeSoi, 1992; Collins, Brown, & Newman, 1989) has become increasingly prominent as a model of instruction. This development is attributable to its potential to help solve the educational problems of brittle skills and inert knowledge that so often arise with traditional schooling (Resnick, 1987; Whitehead, 1929). Recent research in the learning sciences coupled with a shift to the situated cognition paradigm in the cognitive sciences has led to a significant rethinking of the nature of learning and cognition (Brown, et al., 1988; Clancey & Roschelle, 1991) and how we can use technology to support learning (Brown, 1985; Clancey, 1992; Collins, 1991). Cognitive apprenticeship is one manifestation of this new way of thinking about learning.

In the next section of this paper, we describe cognitive apprenticeship as an instructional methodology and explicate its underlying rationale. The following section describes *SMALLTALKER*, a learning environment for Smalltalk programming, and outlines the way in which cognitive apprenticeship has been embodied in the system. We conclude by commenting on the effectiveness of the system in field testing to date and charting the direction in which *SMALLTALKER* will continue to evolve.

Cognitive apprenticeship

The cognitive apprenticeship instructional methodology, as formulated by Collins, Brown, & Newman (1989), consists of six teaching methods: *modeling*, *coaching*, *scaffolding*, *articulation*, *reflection*, and *exploration*. The six methods, in turn, break down into three groups. The first group—modeling, coaching, and scaffolding—is designed to help students acquire an integrated set of cognitive and metacognitive skills through observation and supported practice. The second group—articulation and reflection—is designed to help students gain control of their own problem solving strategies. The final group—exploration—is intended to encourage learner autonomy and problem formulation by the self.

Cognitive apprenticeship embeds the learning of knowledge and skills in their social and functional context. Ir *:nodeling*, an expert performs a task so that students can observe and build a conceptual model of the processes required for task accomplishment. The provision of a conceptual model contributes significantly to success in teaching complex skills without resorting to lengthy practice of isolated subskills. In cognitive domains, unlike traditional apprenticeship settings, modeling often necessitates the externalization of internal cognitive processes. Tacit processes are brought into the open so that students can observe, enact, and practise the requisite skills.

In *coaching*, students are engaged in problem-solving activities that require them to appropriately apply and actively integrate subskills and conceptual knowledge. In this way, conceptual knowledge is exemplified and situated in the contexts of use, thereby grounding the knowledge in experience and making learning meaningful. Consequently, this approach helps to avoid learning outcomes where knowledge remains bound to surface features of problems as they appear in textbooks and is incapable of transfer. The expert coaches students by providing hints, feedback, and reminders, thus assisting them to perform closer to his standard of skill. Coaching requires highly interactive and situated feedback. Hence, the content of coaching interaction is related to specific problems that students face in carrying out a task.

In *scaffolding*, an expert assists students to manage complex task performance. If necessary, he completes those parts of the task that students have not yet mastered. This method may entail students engaging in legitimate peripheral participation (Lave & Wenger, 1991); that is, students participate in the practice of an expert, but only to the degree that they can handle and with the amount of responsibility that they are capable of assuming. Scaffolding is coupled with *fading*, the gradual removal of the expert's support as students learn to manage more of the task on their own. The interplay between observation, scaffolding, and increasingly independent practice aids students in developing the metacognitive skills of self-monitoring and self-correction and in achieving integrated skills and knowledge characteristic of expertise.

In *articulation*, an expert encourages students to explicate their knowledge, reasoning, and problem solving strategies. Such activities provide the impetus for students to engage in the refinement and reorganization of knowledge. The use of synthetic and design tasks in a producer–critic framework is particularly effective in achieving articulation (Allen, 1992; Pea, 1991). Such tasks require students to participate in generating knowledge and evaluating the outcomes of knowledge-building as part of collaborative learning activities. Generative and evaluative processes provide a further basis for concept assimilation and internalization.

In *reflection*, the expert provokes students to compare their problem solving processes with his own, with that of other students, and with an internal cognitive model of the relevant expertise. Such comparisons aid students in diagnosing their difficulties and in incrementally adjusting their performance until they achieve competence. Reflection is facilitated by the provision of abstracted replay that contrasts students' own performance with that of the expert (Collins & Brown, 1988).

In *exploration*, the expert pushes students to be independent learners. Students are only set general goals. At the same time, they are encouraged to identify personal interests and pursue personal goals. Forcing students to engage in exploration teaches them how to frame interesting questions and to identify difficult problems on their own.

The *SMALLTALKER* learning environment

Overview

SMALLTALKER is a Macintosh-based multimedia learning environment for Smalltalk programming. The learning environment supports both concept learning and skill acquisition. A special effort has been made to apply the teaching methods of cognitive apprenticeship in designing and implementing *SMALLTALKER*. When the system is launched, an animation of floating Smalltalk balloons is played repeatedly, together with music, until the mouse is moved. After obtaining basic information about student users, the system leads them through an orientation of the system's interface to allow them to begin interacting effectively with the system. The use of various elements of the interface (eg. mouse, windows, and text manipulation) is demonstrated to students via movie clips. Students are then coached on the use of the interface in a different problem setting.

Having obtained basic interface skills, students are introduced to the important Smalltalk programming concept of message-passing. At this stage, students are taught to view object-oriented programming entirely in terms of objects communicating with one another by message passing. This view is an external view (LaLonde & Pugh, 1990) where programming is treated entirely in terms of message-passing syntax (Shafer & Ritz, 1991). Details of method implementation are hidden. The system's interaction with students is structured so as to reify the relationship between process and product. Thus, representations of objects that students interact with are shown graphically, and message passing invokes changes to objects that are visually representable.

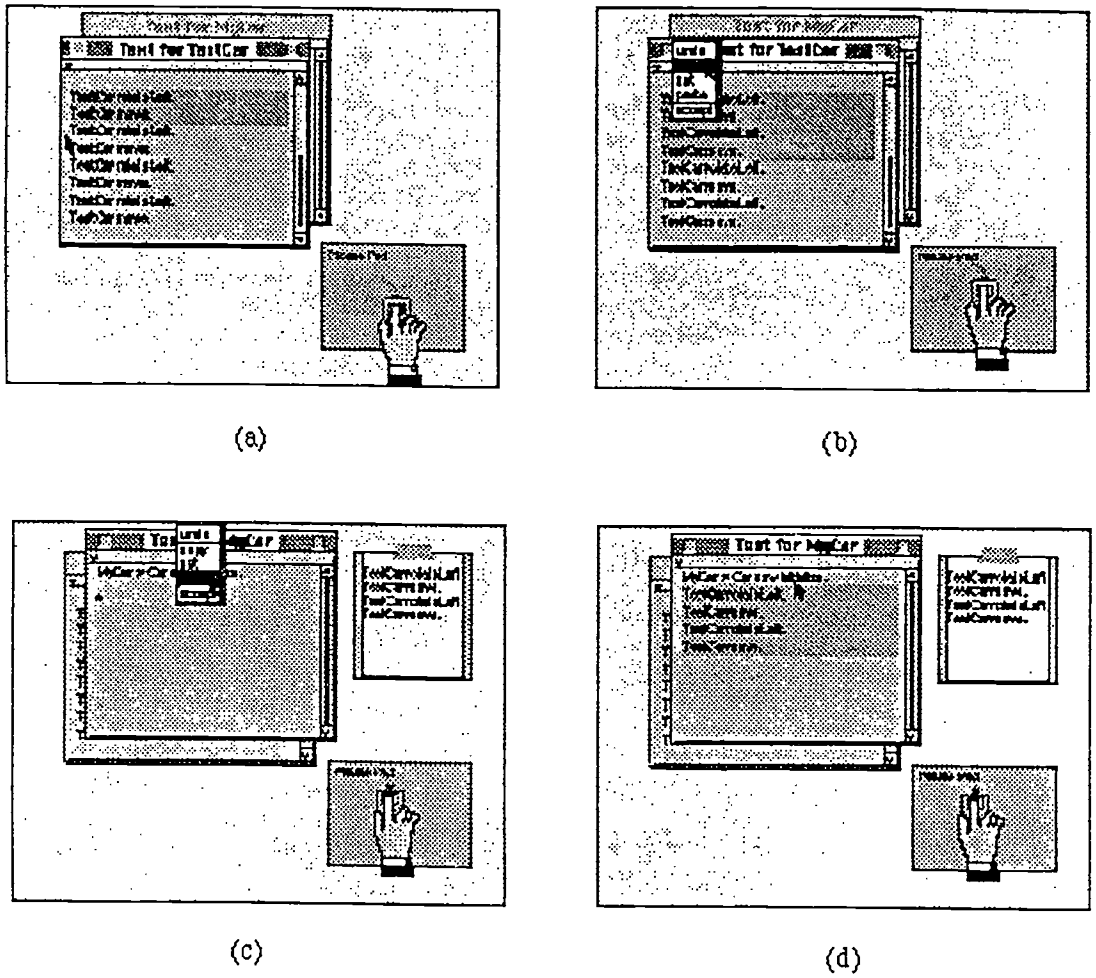
After message passing has been covered, instance methods are introduced. This section opens up the internal view (LaLonde & Pugh, 1990) and its concomitant method-definition syntax (Shafer & Ritz, 1991). The instruction then progresses broadly through the topics of instance variables, classes, class methods, class variables, and superclasses. New classes and new concepts are introduced as and when they are needed in relation to the current goal. In this way, progression through a rigid sequence of topics is avoided. We are currently working on the topic of developing a small application. We plan then to treat the Model–View–Controller paradigm in the context of constructing an authentic application.

In the subsections that follow, we describe how each of the teaching methods has been instantiated in the learning environment.

Modeling

SMALLTALKER is unique in that it fulfills *two* roles: that of instruction presenter and that of coach. In its capacity as instruction presenter, *SMALLTALKER* makes extensive use of modeling. To support enculturation, QuickTime™ movies are used to show an expert presenting information to students. MacroMind Director™ animations are used to present instruction on concepts and skills necessary for Smalltalk programming. The movies are accompanied by the expert's voice narration of the instruction.

Figure 1 provides an illustration of how modeling occurs. The figure comprises four snapshots from a Director animation. In this example, the student is learning about the copy-and-paste technique in text editing. The snapshots show how the expert selects a portion of text (Figure 1a), selects "copy" from the pop-up menu (Figure 1b) resulting in the copied text appearing on a notional clipboard, then selecting "paste" (Figure 1c), thus yielding the pasted text at the insertion point of the selected window (Figure 1d).



(a)

(b)

(c)

(d)

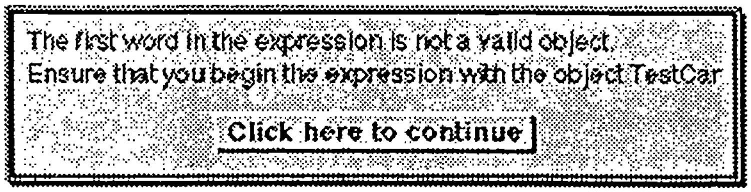
Figure 1 Modeling of copy-and-paste in text editing

Modeling is ideally suited to reifying processes through animation techniques. With the use of multimedia, concepts and processes can be "brought to life," focusing students' attention on the *meaning* and *sense* of what they observe.

In our implementation of modeling, we have framed the learning situation so that students will learn cause-and-effect relationships. Thus, the goals related to a sequence of actions are made explicit *before* the actions are shown, and the results of the actions are shown in a concrete fashion. Our instruction has also been designed to start with concrete and visual objects (eg. TestCar) before moving to more abstract and functional objects (eg. MyRecord) to aid students in grasping the necessary concepts.

Coaching

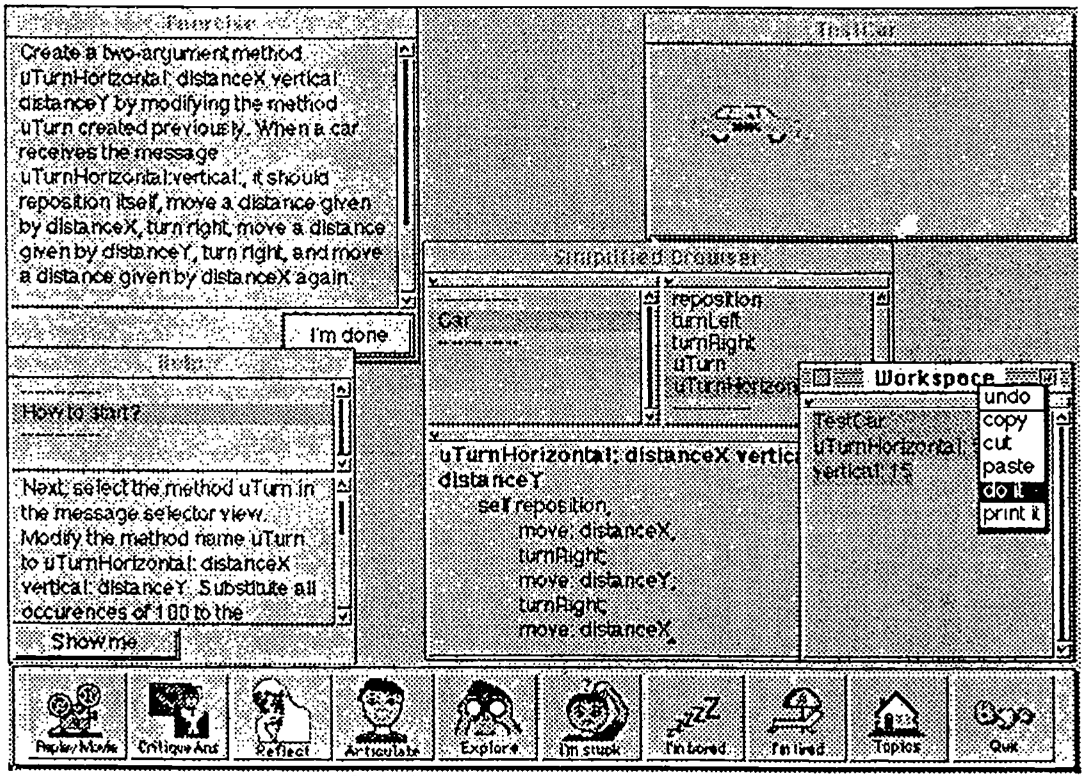
Coaching is a critical activity in any learning environment because it is here that the teacher or system engages in actions to facilitate students' mastery of problem solving. In *SMALLTALKER*, students are made to work on programming problems while they are still being introduced to new concepts and procedures. The most pervasive method of coaching takes the form of feedback to student actions and errors while they are engaged on programming problems. As an example, a dialog box like that shown in Figure 2 will appear if a message expression coded by a student does not begin with the name of an object. Notice that the feedback given is highly situated; it is specific to the particular programming problem being solved.



The first word in the expression is not a valid object.
Ensure that you begin the expression with the object TestCar

Figure 2 A typical feedback dialog

Besides the use of dialogs, feedback is also given by providing visual animation of objects where appropriate. As illustrated in Figure 3, the effect of executing the code written by a student is made self-evident from the animation of the car moving about in the TestCar window. Thus, students are encouraged to form and test hypotheses, diagnose their own performance, and adopt an active attitude to learning. In this way, students are encouraged to develop metacognitive skills.



Create a two-argument method uTurnHorizontal: distanceX vertical: distanceY by modifying the method uTurn created previously. When a car receives the message uTurnHorizontal:vertical:, it should reposition itself, move a distance given by distanceX, turn right, move a distance given by distanceY, turn right, and move a distance given by distanceX again.

How to start?

Next, select the method uTurn in the message selector view. Modify the method name uTurn to uTurnHorizontal: distanceX vertical: distanceY. Substitute all occurences of 100 to the

```
uTurnHorizontal: distanceX vertic
distanceY
     self reposition,
          move: distanceX,
          turnRight,
          move: distanceY,
          turnRight,
          move: distanceX.
```

```
TestCar
uTurnHorizontal:
vertical: 15
```

Figure 3 Snapshot of programming exercise in progress

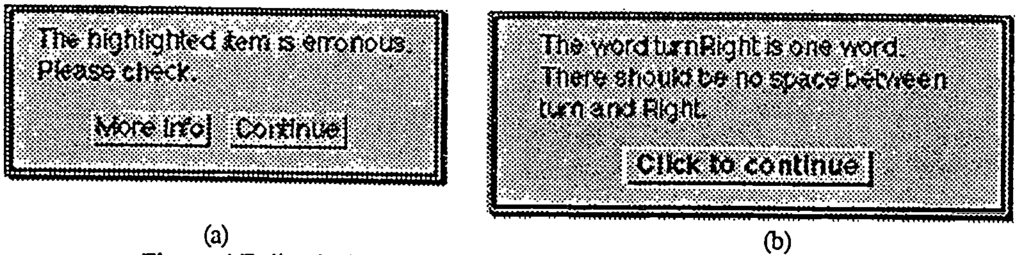
It is important in cognitive apprenticeship that the efforts of students be useful and used. To this end, problem solving exercises often ask students to make enhancements or to add new methods to existing code so that the resulting whole possesses greater value after such changes.

Scaffolding and fading

Scaffolding and fading are difficult teaching methods to implement because they require a teacher to be sensitive tc the detailed needs and difficulties of students engaged in task performance. Our approach to this difficulty has been to place the burden of responsibility on the shoulders of students. *SMALLTALKER* provides a fairly open-ended help system (see Figure 3). Students have the option, while problem solving, to request help by selecting the button "I'm stuck" from the panel of buttons at the bottom of the screen. The help window lists a set of topics on which assistance on specific aspects of the problem students are attempting is available. Making the activation of the help system the responsibility of students encourages them to take active control of their learning. A "Replay Movie" button allows students to review the instructional content related to a problem, whenever they wish.

The "Show me" button in the Help window (see Figure 3) is particularly significant. It functions as a student's last recourse. For this reason, it is separated from the help topic list. We have implemented the "Show me" button such that it only becomes available after the student has selected some percentage of the help items available. This approach prevents students from choosing the easy way out of a problem solving situation. When students eventually select tne "Show me" button, a movie plays, showing how an expert would have solved the programming problem. This action takes us back to the modeling method.

Fading has been implemented in the kind of error trapping that the system engages in. During the course of early problem solving, special checks are incorporated to prevent students from committing errors from which they would be unable to recover. However, as students become more familiar with Smalltalk programming, feedback dialogs are generalized so as to provide less problem solving support for them. Figure 4a illustrates a feedback dialog with fading while Figure 4b illustrates a dialog without fading. However, even with fading operational, students have the option of requesting a more specific dialog if they feel that the level of support received is inadequate in any particular situation. All that they have to do is click on the button "More Info".


The highlighted item is erronous. Please check.

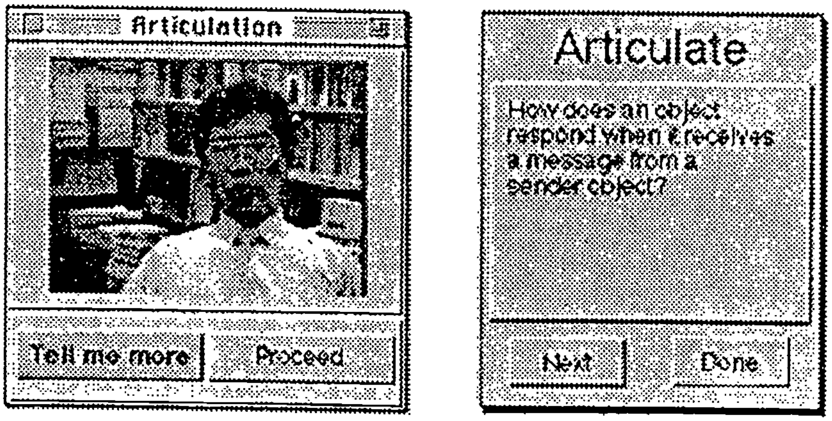
The word turnRight is one word. There should be no space between turn and Right.

(a) (b)

Figure 4 Fading in feedback dialogs. (a) More general. (b) More specific.

Articulation

Computers are ill-suited to handling the natural language articulations of humans. Nevertheless, we attempt to encourage students to articulate their knowledge by posing questions that request students to articulate answers to various conceptual questions, either to themselves or to a friend. When a student clicks the button "Articulate", an "Articulation" window appears (see Figure 5). Clicking on the button "Tell me more" plays a QuickTime movie explaining the purpose of articulation. Clicking "Proceed" takes the student to a specific question the answer to which requires articulation.


Articulate

How does an object respond when it receives a message from a sender object?

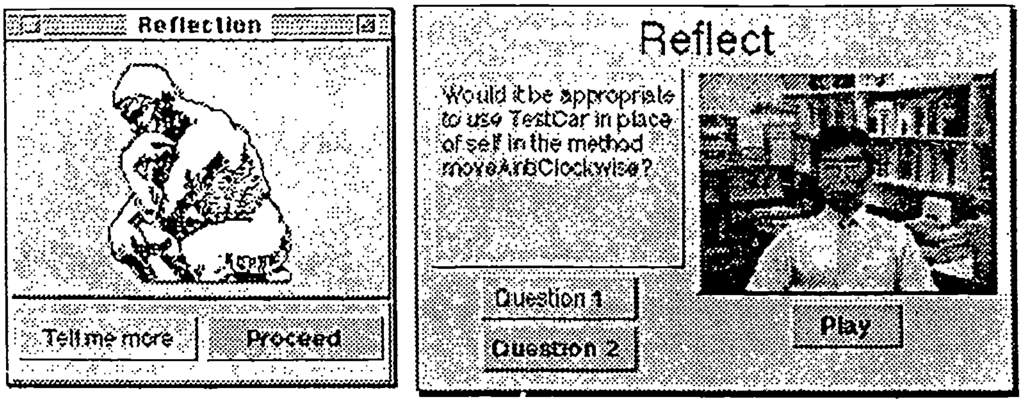
Figure 5 Illustration of the teaching method articulation

Reflection

We attempt to provoke reflection by periodically posing questions that possess deeper conceptual significance. Some reflection questions are included as part of the flow of instruction and, hence, are compulsory. Other questions are optional and can be called up by clicking on the button marked "Reflection". After students have reflected on the question posed, they can click on the button "Play" to listen to an expert expressing his view on the question (see Figure 6). In this way, students can gauge to what extent they have come to appreciate the subject domain in the way that an expert does.

Exploration

It turns out that exploration is an activity that system designers cannot prevent students from engaging in if the system succeeds in gripping their interest and arousing their motivation to learn. We support students'



Would it be appropriate to use TestCar in place of self in the method moveArbClockwise?

Figure 6 Reflection in *SMALLTALKER*

exploration of Smalltalk programming by providing them with an "Explore" button. Clicking on this button takes students into the ParcPlace Smalltalk–80 programming environment. A "Return to *SMALLTALKER*" button allows students to return to *SMALLTALKER*.

Conclusion

Our evaluation of the *SMALLTALKER* implementation to date has been encouraging. In general, students have found the learning environment extremely user-friendly and very supportive of their learning needs. *SMALLTALKER* only attempts to provide students with a first course in Smalltalk programming. Given the complexity of Smalltalk programming and the need to support the development of programming skills as part of a broader community of practice, we have embarked on the development of a complementary learning environment *CALET*, the *C*ognitive *A*pprenticeship *L*earning *E*nvironmen*T*, that has as its objective the provision of situated Smalltalk programming cases for realistic "on-the-job" programming practice.

References

Allen, C. L. (1992). Multimedia learning environments designed with organizing principles from non-school settings. In E. De Corte, M. C. Linn, H. Mandl, & L. Verschaffel (eds.), *Computer-Based Learning Environments and Problem Solving* (pp. 465–484). Berlin: Springer-Verlag.

Brown, J. S. (1985). Idea amplifiers: New kinds of electronic learning environments. *Educational Horizons, 63*, 108-112.

Brown, J. S., Collins, A., & Duguid, P. (1988). *Situated Cognition and the Culture of Learning*. Tech. Rep. No. IRL 88-0008. Institute for Research on Learning.

Burger, M. L., & DeSoi, J. F. (1992). The cognitive apprenticeship analogue: a strategy for using ITS technology for the delivery of instruction and as a research tool for the study of teaching and learning. *International Journal of Man-Machine Studies, 36*, 775-795.

Clancey, W. J. (1992). New perspectives on cognition and instructional technology. In E. Costa (ed.), *New Directions for Intelligent Tutoring Systems* (pp. 3–14). Berlin: Springer-Verlag.

Clancey, W. J., & Roschelle, J. (1991). Situated cognition: How representations are created and given meaning. *Educational Psychologist*, forthcoming.

Collins, A. (1991). Cognitive apprenticeship and instructional technology. In L. Idol & B. F. Jones (eds.), *Educational Values and Cognitive Instruction: Implications for Reform* (pp. 121–138). Hillsdale, NJ: Lawrence Erlbaum.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 1-18). Berlin: Springer-Verlag.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum.

LaLonde, W. R., & Pugh, J. R. (1990). *Inside Smalltalk, Volume 1*. Englewood Cliffs, NJ: Prentice-Hall.

Lave, J., & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. NY: Cambridge University Press.

Pea, R. D. (1991). Learning through multimedia. *IEEE Computer Graphics and Applications, 11*(4), 58–66.

Resnick, L. B. (1987). Learning in school and out. *Educational Researcher, 16*(9), 13-20.

Shafer, D., & Ritz, D. A. (1991). *Practical Smalltalk: Using Smalltalk/V*. NY: Springer-Verlag.

Whitehead, A. N. (1929). *The Aims of Education*. New York: Macmillan.